HOKKAIDO UNIVERSITY

Title	Proper Interval Graphのランダム生成と列挙
Author(s)	齋藤, 寿樹
Description	ERATOセミナ2010 : No.5. 2010年7月2日
Relation	2010年度科学技術振興機構ERATO湊離散構造処理系プロジェクト講究録. p.31-37.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48480
Type	conference presentation
File Information	05_all.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

ERATO セミナ 2010 - No. 05

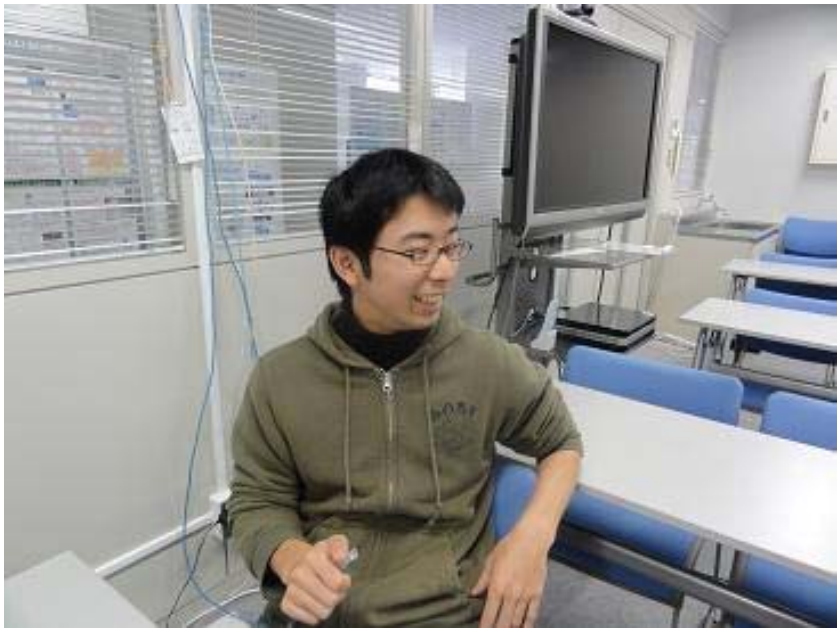
Proper Interval Graph のランダム生成と列挙

齋藤寿樹
JST ERATO 湊離散構造処理系プロジェクト

2010/7/2

概要

Interval Graph は，スケジューリング問題やバイオインフォマティクスなど，いくつもの応用を持つことが知られている．そのため，Interval Graph についての研究はいくつも存在し，また Interval Graph 上のアルゴリズムが開発されている．実際の応用で開発されたアルゴリズムを適用する場合，そのアルゴリズムは理論的に効率がよいだけでなく，実装上効率がよいことを実験的解析によって示す必要がある．実験的に解析をするとき，テストデータに偏りがある場合，正しい解析結果を得ることができないため，偏りのないテストデータを生成する必要がある．本発表では，Interval Graph の部分クラスである Proper Interval Graph をランダム生成や列挙するアルゴリズムを提案する．ランダム生成アルゴリズムは数え上げを用いたアルゴリズムで，n 頂点の連結な Proper Interval Graph を一様ランダムに生成する．列挙アルゴリズムは逆探索に基づいたアルゴリズムで，n 頂点の連結な Proper Interval Graph を漏れなく，重複なく出力する．



斎藤研究員

Random Generation and Enumeration of Proper Interval Graphs

Toshiki Saitoh (ERATO)

Joint work with

Katsuhisa Yamanaka, Masashi Kiyomi, and Ryuhei Uehara

第5回ERATOセミナー
2010年7月2日

Motivation

- Generate an object *uniformly at random*
- Enumerate *all* object

Computer experiment

We want suitable test data …

- Input graphs
  - Permutation graphs
  - Interval graphs

→ Bipartite permutation graphs

→ Proper interval graphs (P.I.G)

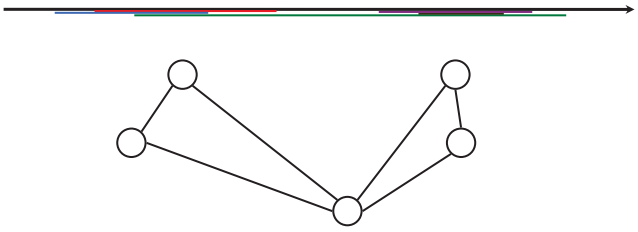
Random generation
Enumeration

Our Algorithms

- Random Generation
  - Input: Natural number n
  - Output: Connected P. I. G. of n vertices
    - Uniformly at random
    - Using a counting algorithm
    - $O(n+m)$ time (m: #edges)
- Enumeration
  - Input: Natural number n
  - Output: All the connected P. I. G. of n vertices
    - Without duplication
    - Based on reverse search algorithm
    - O(1) time/graph

Interval Graphs

- Have interval representations

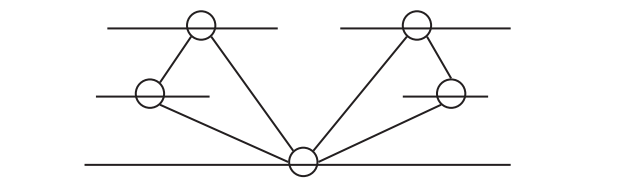


Proper Interval Graphs

- Have unit interval representations

Every interval has the same length

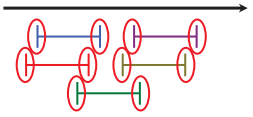
String representations



Definition

- String Representation
  - Encodes a unit interval representation by a string
    - Sweep the unit interval representation from left to right
      - Left endpoint → "(" : left parenthesis
      - Right endpoint → ")" : right parenthesis

Right endpoints appear in order of their left endpoint appearances



Unit Interval Representation

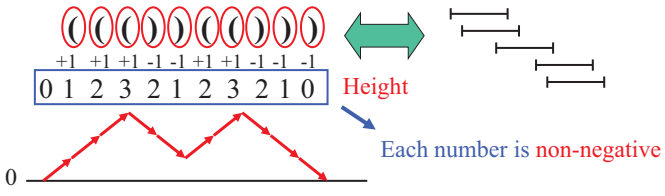
((())(()))

String Representation

Height = # “(” - # “)”

String Representation

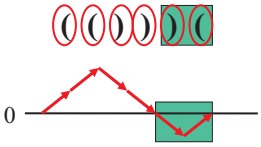
- **Property of string rep. of P. I. G. of *n* vertices**
  - Number of parentheses: $2n$
    - Number of “(”: n Number of “)”: n
  - Non-negative
    - Each left parenthesis exists in the left side of its right parenthesis



String Representation

- **Property of string rep. of P. I. G. of *n* vertices**
  - Number of parentheses: $2n$
    - Number of “(”: n Number of “)”: n
  - Non-negative
    - Each left parenthesis exists in the left side of its right parenthesis

Negative height



?

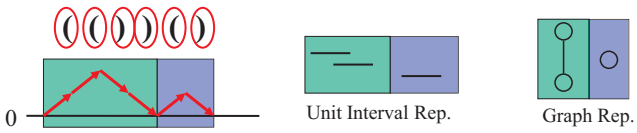
Not correspond to any interval rep..

String Representation

- **Property of string rep. of P. I. G. of *n* vertices**
  - Number of parentheses: $2n$
    - Number of “(”: n Number of “)”: n
  - Non-negative
    - Each left parenthesis exists in the left side of its right parenthesis

Each component corresponds to each area that is bounded by 2 places whose heights are 0.

There are more than 2 places whose heights are 0.

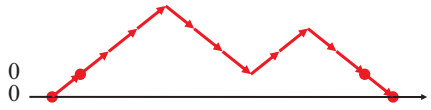


String Representation

- Observation 1
  - String rep. of connected P. I. G.
    - Have exactly 2 places whose heights are 0.
      - The left end and the right end

The string excepted both ends parentheses is non-negative

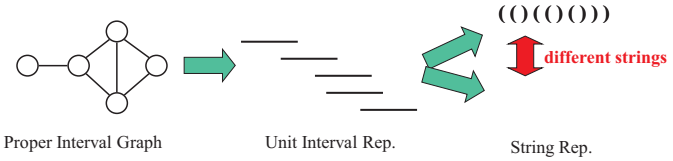
(((( ))) (( ))))



String Representation

- Lemma 1. (X. Dell, P. Hell, J. Huang, 1996)
  - A connected P. I. G. has only one or two string rep.

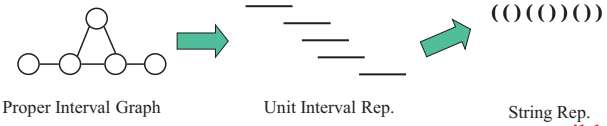
This graph has only two string representations.



String Representation

- Lemma 1. (X. Dell, P. Hell, J. Huang, 1996)
  - A connected P. I. G. has only one or two string rep.

This graph has only one string representation.



reversible

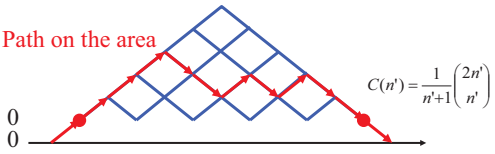
Random Generation Algorithm of Proper Interval Graphs

Random Generation Algorithm

- Generate a string rep. uniformly at random
  - Using a counting algorithm
    - (Generalized) Catalan number

(((( )) ( ) ( )))

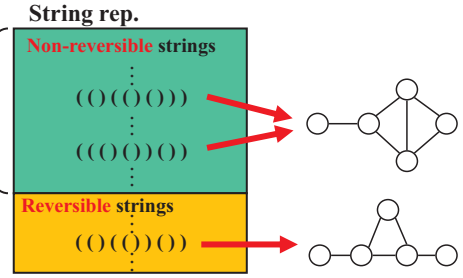
String rep. : Path on the area



Adjust the Generation Probability

Not easy

Decrease the generation probability



A generation probability of a graph corresponding to non-reversible strings is higher than that of reversible one

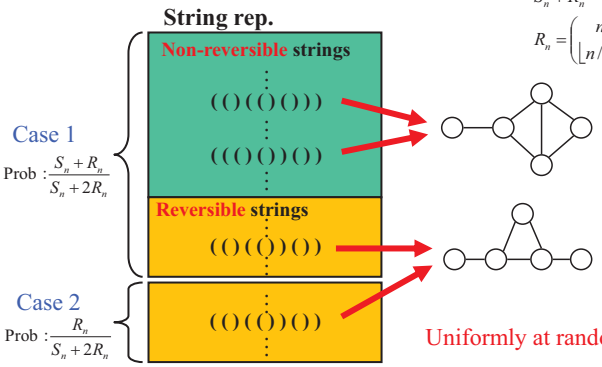
S_n: # non-reversible strings

R_n: # reversible strings

Adjust the Generation Probability

$S_n + R_n = C(n)$

$R_n = \binom{n}{\lfloor n/2 \rfloor}$



Uniformly at random

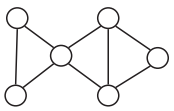
Generalized Catalan Number

$C(n,i) = \frac{i}{2n+i}\binom{2n+i}{n}$

Case 1

- Generation of a string uniformly at random
  - Generate parentheses from left
    - Select "(" or ")"

((( )) (( ) ( )))

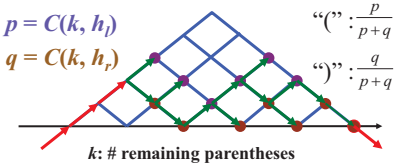


$p = C(k, h_l)$

$q = C(k, h_r)$

"(" : $\frac{p}{p+q} = \frac{h(k+h+2)}{2k(h+1)}$

")" : $\frac{q}{p+q} = \frac{(k-h)(h+2)}{2k(h+1)}$



k: # remaining parentheses

h: Height

Time complexity

- String Rep. : $O(n)$
- Graph Rep. : $O(n+m)$

m: # edges

Generalized Catalan Number

$C(n,i) = \frac{i}{2n+i}\binom{2n+i}{n}$

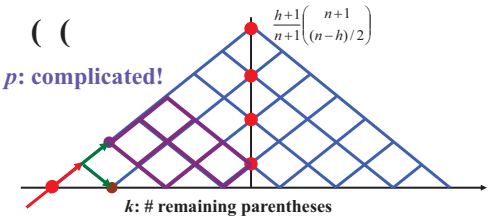
Case 2

Symmetric

- Generation of reversible string uniformly at random
  - Generate a half of the string
  1. Choose the height at the center
  2. Generate parentheses from the left end
     - Select "(" or ")"

((

p: complicated!



k: # remaining parentheses

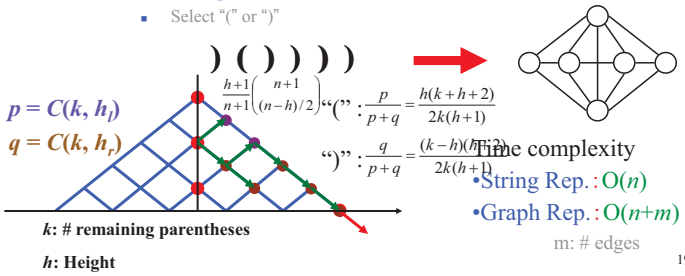
h: Height

Generalized Catalan Number

$$C(n,i)=\frac{i}{2n+i}\binom{2n+i}{n}$$

Case 2

- Generation of reversible string uniformly at random
  - Generate a half of the string from the center to the right end
  1. Choose the height at the center
  2. Generate parentheses from the center
     - Select "(" or ")"

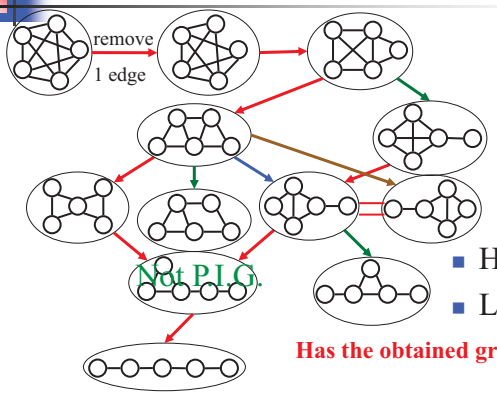


Time complexity
- String Rep. : O(*n*)
- Graph Rep. : O(*n*+*m*)

m: # edges

Enumeration Algorithm of Proper Interval Graphs

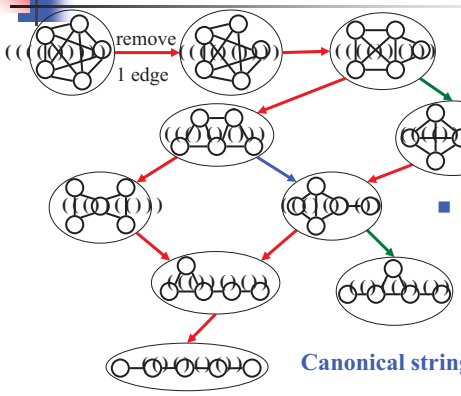
Simple Enumeration Algorithm



- Huge memory
- Low speed

Has the obtained graph outputted?

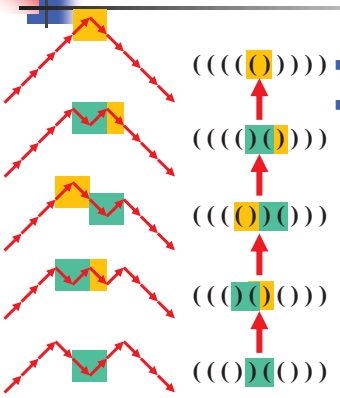
Reverse Search Algorithm



- Spanning Tree
  - Parent-child
- Our algorithm
  - Canonical string
  - O(1) time/graph
  - O(n) space

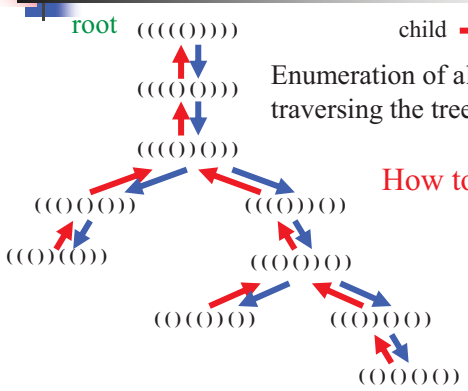
Canonical string x ≦ reverse of the x

Parent-Child Relation



- Root: ((… ( )) …)
- Parent of canonical string x
  - Replace the leftmost ") (" of x with "( )"
    - Parent of x is canonical
    - The root is the ancestor of x

Tree (n=5)

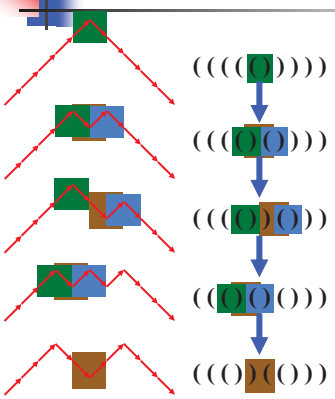


Enumeration of all the strings by traversing the tree from the root

How to find children?

- Parent of canonical string x
  - Replace the leftmost "**)** **(**" of x with "**(** **)**"

Candidates of Children

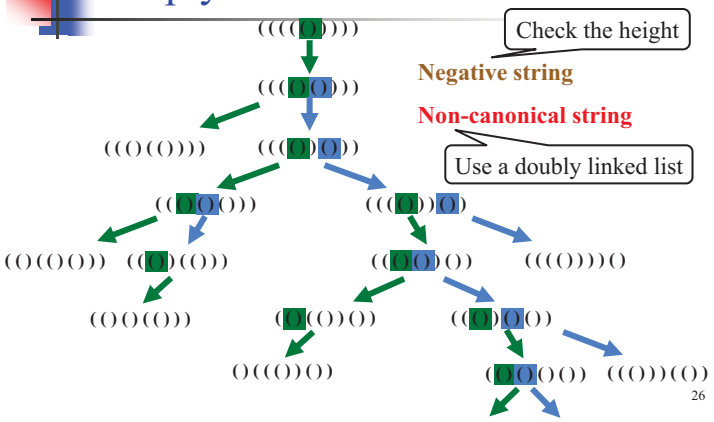


- Child of canonical string y
  - Replace "( )" of y with ") ("
    1. The leftmost "( )", or
    2. If the parenthesis after the leftmost ") (" is ")" then "( )" can be the candidate
- Next candidates
  - **Neighbors** of the **replaced parentheses**

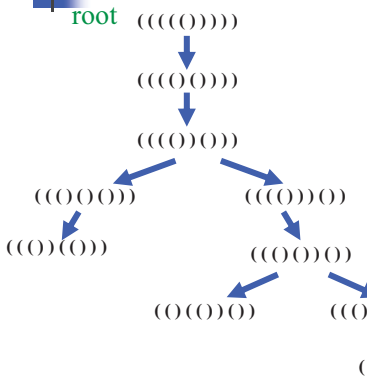
Find candidates: O(1) time

Simply Execution

Check child: O(1) time



Tree (n=5)



Find child: O(1) time
Output only the differences
Prepostorder manner

Output string: O(1) time

Output P.I.G.: O(1) time

Conclusion and Future Work

- Random Generation and Enumeration of Connected Proper Interval Graphs of n vertices
  - Random Generation : O(n+m) time
  - Enumeration : O(1) time/graph
  - n vertices ⇒ at most n vertices
- Random Generation and Enumeration of Interval Graphs